

1
2
3 **METHOD AND SYSTEM FOR IDENTIFYING**
4 **PROGRAM MODULE FUNCTIONALITY NEEDED BY**
5 **A COMPUTER WHEN DISCONNECTED FROM A**
6 **NETWORK**

7
8 **Technical Field**

9 This invention relates to identifying program module
10 functionality needed by a computer. More particularly, this
11 invention relates to a method and system for identifying the
12 program module functionality needed by a computer when
13 disconnected from a network and storing this program module
14 functionality on the computer's hard drive.

15
16 **Background of the Invention**

17 Mobile or laptop computing has become more
18 popular as mobile computers have decreased in price and
19 increased in performance. Many mobile computer users use their
20 computers to connect to a network at the office. These same users
21 may disconnect from the network to use their mobile computers
22 when away from the office. Mobile computing demands that
23 users have access in a disconnected environment to the data and
24 the applications that are typically available in a connected
25 environment, i.e., when connected to a network. While preparing
26 for offline use, users generally think in terms of documents, not
27 in terms of applications. Mobile computers do not provide an
28 intelligent list of applications which may be needed when the
29 mobile computer is disconnected from the network.

30 As mobile computing becomes prevalent, the
31 transition between network-connected use and offline use should
32 be transparent, quick and painless. A mobile computer user
33 should not have to worry about the management of applications
34 and documents on her laptop. The mobile computer user needs to
35 make sure that the documents and applications required when the

1 unit is offline are on the mobile computer before disconnecting it.
2 Thus, there is a need for a method and system for managing the
3 documents and files that are needed on a mobile computer when
4 the mobile computer is disconnected from a network.

5 However, having needed files and documents on a
6 mobile computer does not mean that the application program
7 functionality needed to run these documents is locally available,
8 i.e. stored on the mobile computer. Thus, there is a need for a
9 method and system for intelligently identifying a list of documents
10 the user may need when offline and mapping the documents to the
11 necessary application program functionality needed to execute the
12 documents.

13 Thus, given a set of documents, there is a need for a
14 method and system for mapping the set of documents to a set of
15 application program functionality required to run the set of
16 documents. There is a further need for a method and system for
17 a method and system for intelligently identifying a list of
18 documents that may be needed by a user when off-line, i.e.,
19 disconnected from a network.

20 21 **Summary of the Invention**

22 The present invention satisfies the above described
23 needs by providing a method and system for identifying the
24 program application functionality needed by a computer when
25 disconnected from a network and storing this program module
26 functionality on the computer.

27 In one aspect, the invention identifies a handler
28 routine for each file saved to a local computer, or marked to be
29 available off-line, and sending each file to the identified handler
30 routine. The handler routine may then determine the application
31 program functionality required to execute each file, i.e., read and
32 edit a file. The application program functionality may comprise
33 products, features and components as defined below in the
34 detailed description.

1 In another aspect, the invention identifies a handler
2 routine for each file in the set of files by identifying a type for the
3 file by mapping a file extension for each file to a class
4 identification. Then, for each file in the set of files, the class
5 identification is mapped to a handler routine and each file is sent
6 to the mapped handler routine.

7 In still another aspect, the present invention
8 comprises a document identification engine (DIE) for creating a
9 list of files stored locally on a computer. The DIE sends the list
10 of files to a document-mapping engine (DME), which identifies a
11 proper handler routine for each file in the list of files. The DME
12 then sends each file to the proper handler routine(s). The handler
13 routine(s) identifies the application program functionality needed
14 to execute each file and sends a list of needed application
15 functionality to the DME or a migration engine (ME). The ME
16 determines the current status of the needed application
17 functionality. If the status of the needed application functionality
18 indicates that the needed application program functionality is not
19 installed locally on the computer, then the ME may install the
20 needed application program functionality to the computer.

21 These and other features, advantages, and aspects of
22 the present invention may be more clearly understood and
23 appreciated from a review of the following detailed description of
24 the disclosed embodiments and by reference to the appended
25 drawings and claims.

26 **Brief Description of the Drawings**

27 Fig. 1 is a block diagram of a computer that provides
28 the exemplary operating environment for the present invention.

29 Fig. 2 is a block diagram of typical program modules
30 that may be included in an exemplary embodiment of the present
31 invention.

32 Fig. 3 is a flowchart illustrating a method for
33 identifying application program features needed when a computer
34

1 is offline in accordance with an embodiment of the present
2 invention.

3 Fig. 4 is a flowchart illustrating a method for
4 identifying application program features needed when a computer
5 is offline in accordance with another embodiment of the present
6 invention.

7

8 **Detailed Description**

9 The present invention is directed to a method and
10 system for identifying the application program functionality that
11 may be needed to use a document, or file, when disconnected
12 from a network environment. In one embodiment, the operating
13 system may automatically identify the documents that will likely
14 be needed by the user when his computer is disconnected from the
15 network. The invention may be incorporated into an operating
16 system program module. Briefly described, the operating system
17 allows a user to select the documents, or files, that will be needed
18 by the user when his computer is disconnected from the network.
19 It should be understood that different parts of the operating
20 system, and even other applications, may perform the steps of the
21 invention described herein. In one embodiment, the invention
22 identifies the application program functionality that will be
23 needed to run the documents selected by the user when the
24 computer is disconnected from the network. The present
25 invention may also identify whether this application program
26 functionality is stored locally on the computer, and, if not, the
27 present invention may store this application program functionality
28 locally on the computer.

29 Having briefly described embodiments of the present
30 invention, an exemplary operating environment for the present
31 invention is described below.

32

33 **Exemplary Operating Environment**

34 Referring now to the drawings, in which like
35 numerals represent like elements throughout the several figures,

5552260

1 aspects of the present invention and the exemplary operating
2 environment will be described.

3 Fig. 1 and the following discussion are intended to
4 provide a brief, general description of a suitable computing
5 environment in which the invention may be implemented. While
6 the invention will be described in the general context of an
7 operating system that runs in conjunction with a personal
8 computer, those skilled in the art will recognize that the invention
9 also may be implemented in combination with other program
10 modules. Generally, program modules include routines,
11 programs, components, data structures, etc. that perform
12 particular tasks or implement particular abstract data types.
13 Moreover, those skilled in the art will appreciate that the
14 invention may be practiced with other computer system
15 configurations, including hand-held devices, multiprocessor
16 systems, microprocessor-based or programmable consumer
17 electronics, minicomputers, mainframe computers, and the like.
18 The invention may also be practiced in distributed computing
19 environments where tasks are performed by remote processing
20 devices that are linked through a communications network. In a
21 distributed computing environment, program modules may be
22 located in both local and remote memory storage devices.

23 With reference to Fig. 1, an exemplary system for
24 implementing the invention includes a conventional personal
25 computer **20**, including a processing unit **21**, a system memory
26 **22**, and a system bus **23** that couples the system memory to the
27 processing unit **21**. The system memory **22** includes read only
28 memory (ROM) **24** and random access memory (RAM) **25**. A
29 basic input/output system **26** (BIOS), containing the basic routines
30 that help to transfer information between elements within the
31 personal computer **20**, such as during start-up, is stored in ROM
32 **24**. The personal computer **20** further includes a hard disk drive
33 **27**, a magnetic disk drive **28**, e.g., to read from or write to a
34 removable disk **29**, and an optical disk drive **30**, e.g., for reading
35 a CD-ROM disk **31** or to read from or write to other optical

860221 5652260

media. The hard disk drive 27, magnetic disk drive 28, and optical disk drive 30 are connected to the system bus 23 by a hard disk drive interface 32, a magnetic disk drive interface 33, and an optical drive interface 34, respectively. The drives and their associated computer-readable media provide nonvolatile storage for the personal computer 20. Although the description of computer-readable media above refers to a hard disk, a removable magnetic disk and a CD-ROM disk, it should be appreciated by those skilled in the art that other types of media which are readable by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, and the like, may also be used in the exemplary operating environment.

A number of program modules may be stored in the drives and RAM 25, including an operating system 35, one or more application programs 36, installer program module 37, program data 38, and other program modules (not shown).

A user may enter commands and information into the personal computer 20 through a keyboard 40 and pointing device, such as a mouse 42. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 21 through a serial port interface 46 that is coupled to the system bus, but may be connected by other interfaces, such as a game port or a universal serial bus (USB). A monitor 47 or other type of display device is also connected to the system bus 23 via an interface, such as a video adapter 48. In addition to the monitor, personal computers typically include other peripheral output devices (not shown), such as speakers or printers.

The personal computer 20 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 49. The remote computer 49 may be a server, a router, a peer device or other common network node, and typically includes many or all of the

1 elements described relative to the personal computer 20, although
2 only a memory storage device 50 has been illustrated in Fig. 1.
3 The logical connections depicted in Fig. 1 include a local area
4 network (LAN) 51 and a wide area network (WAN) 52. Such
5 networking environments are commonplace in offices, enterprise-
6 wide computer networks, Intranets and the Internet.

7 When used in a LAN networking environment, the
8 personal computer 20 is connected to the LAN 51 through a
9 network interface 53. When used in a WAN networking
10 environment, the personal computer 20 typically includes a
11 modem 54 or other means for establishing communications over
12 the WAN 52, such as the Internet. The modem 54, which may be
13 internal or external, is connected to the system bus 23 via the
14 serial port interface 46. In a networked environment, program
15 modules depicted relative to the personal computer 20, or
16 portions thereof, may be stored in the remote memory storage
17 device. It will be appreciated that the network connections shown
18 are exemplary and other means of establishing a communications
19 link between the computers may be used.

21 Discussion of Terminology

22 Before discussing Fig. 2, a brief discussion of
23 terminology is needed. In accordance with an exemplary
24 embodiment of the present invention, the installer program
25 module 37 recognizes three principal elements: products, features
26 and components. The installer program module 37 is also
27 described in co-pending application serial no. _____, entitled
28 "Use of Relational Databases for Software Installation", which is
29 assigned to the same assignee, filed on September 21, 1998, and
30 incorporated by reference herein.

31 A "product" represents an entire application
32 program, such as the "MICROSOFT OFFICE" application
33 program marketed by Microsoft Corporation of Redmond,
34 Washington. Each product has a globally unique identifier known

860521 5652260

See
B

1 as a Product Code which allows each product to be distinguished.
2 Each product is made up of one or more features.

3 A feature is a granular piece of the product that a
4 user may choose to install. Features typically correspond roughly
5 to the functional features of the product itself, such as a "Proofing
6 Tools" feature or a "WORD" feature. Each feature is essentially
7 a grouping of components and may also include other features.
8 Features need not be globally unique, and therefore may be
9 identified by any appropriate means, such as with a textual feature
10 identifier.

11 A component is a collection of resources, such as
12 files or registry keys, that are all installed or uninstalled as a unit.
13 Components are the building blocks of the product that are not
14 exposed to the user. A resource, such as a file or a registry key,
15 may be part of only one component. Two components may not
16 share the same resource whether they are part of the same
17 product or parts of different products. Each component has a
18 globally unique identifier known as a Component Code. One
19 resource within the component is designated as a key file. The
20 key file may be any resource, such as a file or registry key,
21 within the component.

22 As used herein, application program functionality
23 will be used to refer to products, features and components.

24 25 **Identifying Needed Files and Application Program** 26 **Functionality**

27 As mentioned above in the Background, application
28 program modules do not currently provide a management tool
29 that allows users to identify their off-line application needs, and
30 synchronize their data with the applications required to use the
31 data. In addition, data formats are becoming increasingly
32 complex and users need to have a way to manage complex data
33 formats, such as OLE structured storage with embedded OLE
34 objects or HTML pages with multiple links. Moreover, as
35 described above in the Terminology section, some application
36 program modules now have functionality that may be installed on-

2025 RELEASE UNDER E.O. 14176

7 The invention is a system and method for identifying
8 a list of documents, or files, and application program module
9 functionality that a user may need when the computer is
10 disconnected from a network. The invention may be used with
11 laptop computers or desktop computers which are connected to a
12 network.

Referring now to Fig. 2, a block diagram of typical program modules that may be included in an exemplary embodiment **200** of the present invention is illustrated. Ensuring that the appropriate documents and the required application program functionality are available offline may begin by choosing documents to be made available offline. This is typically performed via a Document Identification Engine (DIE) **205**.

- The “My documents” folder;
- Recently used documents;

- 1 • Documents and folders that the user has specifically marked as
- 2 “Need when off-line”. For example, every time the user
- 3 creates/publishes a document, he or she can mark it as “Need
- 4 when off-line”;
- 5 • The “Desktop” folder; and
- 6 • Dependent files, e.g., links and embeddings in a document,
- 7 macros that are associated with command bars, etc.

8 Given the above set of document locations, or a
9 similar set, the DIE **205** may yield a list of documents required
10 for offline use. Multiple DIEs may be required due to different
11 types of storage (for example, web servers, file servers, MAPI
12 stores, etc.).

13 This list generated by the DIE **205** is then collected
14 by a Document-Mapping Engine (DME) **210**. The DME **210**
15 uses the list to determine which functionalities and applications the
16 user requires. The DME identifies document classes and compiles
17 a list of class identifications.

18 The DME **210** may yield a set of program module
19 functionalities which is required for the selected offline
20 documents. For each document in the list furnished by the DIE
21 **205**, the DME **210** may identify the class ID of the type of
22 document based on document extensions. The DME **210** may
23 also identify the class ID of the type of document by using more
24 than document extensions. For example, for OLE-composed
25 documents, the class ID is actually stored in the file itself. It may
26 then compile this into a list of class identifications to be handed
27 off to the specific document handlers.

28 As mentioned above, the Document Mapping Engine
29 (DME) **210** may identify the class IDs for each document
30 identified by the Document Identification Engine (DIE) **205**.
31 However, it should be understood that mapping a document to a
32 proper handler routine may be performed several different ways,
33 such as by using a document extension, etc. Each document will
34 then be handed to a handler **215** specific to its document type.
35 The handler may then map the file to product and feature

1 identifications known by the installer program module **37**. As
2 mentioned, specific document handlers **215** may perform this
3 mapping of document class identifications. Each different handler
4 may return product and feature mappings for a specific document
5 type. For example, one or more OFFICE handlers may
6 understand OFFICE file formats and be able to map format
7 contents to specific OFFICE features.

8 The handler **215** may identify the product and
9 features necessary for the specified document, and return the
10 required product and features to the DME **210**. The DME will
11 collect all of the product and feature identifications for all of the
12 documents, and may sort them according to frequency of the
13 occurrence of any given product and feature ID. However, the
14 handlers themselves may also return importance rankings instead
15 of leaving the decision of importance to the DME. The product
16 and feature identifications are then sent to a Migration Engine
17 **220** (ME).

18 The Migration Engine (ME) **220** may be able to
19 identify the application functionality that is most critical given the
20 document types that are most prevalent. This list will then be
21 used by the Migration Engine to install the necessary application
22 functionality. It should also be understood that decisions could be
23 made based on factors other than which document types are most
24 prevalent.

25 In the event that the ME determines that it cannot
26 install all the requested application functionality due to a lack of
27 disk space, it will return the list of documents to the DIE. The
28 DIE will then present the user with an interface that allows them
29 to modify their selection of documents.

30 Referring now to Fig. 3, a flowchart illustrating a
31 method **300** for identifying application program functionality
32 needed when a computer is offline in accordance with an
33 embodiment of the present invention will be described. The
34 method begins at start step **305** and proceeds to step **310**. At step
35 **310**, the files, or documents, to be stored locally on the computer

1 is determined. It should be understood that the process of
2 determining the files to be stored locally on the computer may be
3 a manual process, such as the user storing files locally on the
4 computer. It should also be understood that the process of
5 determining the files to be stored locally on the computer may be
6 an automatic process. For example, the operating system may
7 include a set of rules for determining a default list of folders and
8 files the user will need when offline.

9 After the files to be stored locally on the computer
10 are determined at step **310**, the method proceeds to step **315**.
11 The types of files stored locally on the computer are identified
12 and the files are sent to a proper handler routine at step **315**. For
13 example, all WORD files stored locally on the computer are sent
14 to a handler that understands WORD documents. The method
15 then proceeds to step **320**.

16 At step **320**, the application program functionality
17 needed to execute the files is identified by the handler routine(s).
18 The method **300** then proceeds to step **325**.

19 At step **325**, the application program functionality
20 needed, and not already installed locally, is installed locally on the
21 computer. The method ends at step **399**.

22 Referring now to Fig. 4, a flowchart illustrating a
23 method **400** for identifying application program functionality
24 needed when a computer is offline in accordance with an
25 embodiment of the present invention will be described.

26 The method **400** begins at start step **405** and
27 proceeds to step **410** where a Document Identification Engine
28 (DIE) determines the files, or documents, to be stored on the local
29 computer and the types of files that are to be stored locally (such
30 as .doc, .htm, .xls). The method **400** then proceeds to step **415**.
31 The DIE may be triggered when the user shuts down the
32 computer or undocks the computer. The DIE may also be
33 triggered manually by the user from the control panel, start
34 menu, or otherwise.

1 At step **415**, the DIE creates a list of files
2 determined at step **410** and these files are stored locally to the
3 computer if they are not already stored locally. The method **400**
4 then proceeds to step **420**.

5 At step **420**, the list of files created by the DIE at
6 step **415** is transferred to a Document Mapping Engine (DME).
7 The method **400** then proceeds to step **425**.

8 At step **425**, the DME identifies a handler routine
9 for each file. It should be understood that the DME may identify
10 the type of files by mapping the file extension (such as .doc, .http,
11 .xls) to a class identification. It should be understood that each
12 class identification may be associated with a handler routine. It
13 should also be understood that the proper handler routine may be
14 identified by other means, such as by OLE-compound document
15 CLSID. The method **400** then proceeds to step **430**.

16 At step **430**, the DME sends each file to its proper
17 handler routine for processing. The method **400** then proceeds to
18 step **435**. At step **435**, each handler routine is used to identify
19 specific needs and requirements for a particular file. For
20 example, certain application program functionality may be needed
21 to execute a particular file while other application program
22 functionality may be needed to execute other files. Each handler
23 routine maps a file itself to the application program features that
24 need to be installed for the file to execute. The method **400** then
25 proceeds to step **440**.

26 At step **440**, the handler routines return the
27 application program functionality needed for each file to the
28 DME so that the DME has a complete list of all the application
29 program functionality needed to execute the saved local files. The
30 method then proceeds to step **445**. It should be understood that it
31 is possible that the DME will need to send one or more new files
32 to a handler routine in response to the instructions from the
33 handler routines. For example, an EXCEL spreadsheet may
34 contain an embedded WORD document. In that case, the EXCEL
35 handler routine may not recognize the WORD document and may

1 return the WORD document to the DME to determine the proper
2 handler routine for the WORD document. The DME would then
3 transmit the WORD document to the proper handler.

At step **445**, the DME **210** transmits the list of needed application program functionality to a migration engine (ME) **220**. The method then proceeds to step **450**. At step **450**, the ME **220** determines the current status of the application program functionality, i.e., whether the functionality is available and installed locally. The method then proceeds to step **455**.

At step 455, any application program functionality that is not installed locally is installed to the local computer, if it is available. The method 400 then ends at step 499.

Thus, from the foregoing description, it will be apparent to those skilled in the art that the present invention provides a method and system for identifying a set of application program functionality that may be needed on a computer by a user when the computer is disconnected from a network environment, or when a computer does not have a CD-ROM connection. For example, a home user may have a laptop computer with a docking station, but not have the computer connected to a network. When at home, the user can use the CD-ROM drive attached to the docking station to run application bits "from the source." However, when disconnected from the docking station, the laptop computer has no CD-ROM drive and the user will need the bits to be local.

It should be understood that the storing of the application functionality to the local computer may be done on a priority basis to deal with storage constraints. For example, if six word processing documents are stored locally and one spreadsheet document is stored locally, then the application functionality for the word processor should be stored to the local computer before attempting to store the spreadsheet functionality. It should also be understood that the user may be presented with the option of determining which applications need to be stored locally if there are storage constraints.

1 It should also be understood that if a single file
2 comprises other types of files, then the handler will be able to
3 identify these other type files or send them back to the DME so
4 that they may be sent to a proper handler routine. For example,
5 if a word processing document comprises a spreadsheet document,
6 then the handler will be able to identify these types of files and
7 send them to their proper handler. Thus, the present invention
8 will be able to identify when a single document needs multiple
9 application functionality. This concept may be referred to as
10 embedded mapping. Thus, for any sub-part of a file (such as an
11 OLE sub-storage or a hyperlink), the handler may send the sub-
12 part back to the DME which may try to map it to a different
13 handler.

14 It will also be understood that alternative
15 embodiments will become apparent to those skilled in the art to
16 which the present invention pertains without departing from its
17 spirit and scope. Accordingly, the scope of the present invention
18 is defined by the appended claims rather than the foregoing
19 description.

09223595-12098
SECRET-96922260